Supplemental code file for manuscript titled: Human Milk Oligosaccharide Utilization in Intestinal Bifidobacteria is Governed by a Master Transcriptional Regulator NagR

Aleksandr A. Arzamasov *Aleksandr A. Arzamasov, Aruto Nakajima, Mikiyasu Sakanaka, Miriam Ojima*

Contents

1 Background

Insert background text

The code below describes: 1. Analysis of RNA-seq data: processing of raw fastq files and mapping reads to the *Bifidobacterium longum* subsp. *infantis* ATCC 15697 transcriptome 2. Analysis of EMSA data: processing of gel quantification data and building of 4-PL models

2 Reproducibility and accessibility

To reproduce **all** steps listed below, you will need to download raw fastq files from the Gene Expression Omnibus, under accession instert_accession. All code used in this analysis, including the Rmarkdown document used to compile this supplementary code file, is available on GitHub **here**. Once the GitHub repo has been downloaded, navigate to `NagR_manuscript/` to find the Rmarkdown document as well as an RProject file. This should be your working directory for executing code. Downloaded fastq files should be placed in: `NagR_manuscript/data/fastq/`.

---

3 R packages used

A set of R packages was used for this analysis. All graphics and data wrangling were handled using the tidyverse suite of packages. All packages used are available from the Comprehensive R Archive Network (CRAN), Bioconductor.org, or Github.

```
library(tidyverse)
library(tximport)
library(gt)
library(edgeR)
library(matrixStats)
library(cowplot)
library(ggrepel)
library(pheatmap)
```

---

4 External R functions used

A set of external R functions was used to keep the code tidy. All used R scripts with functions can be found in `NagR_manuscript/code/`.

```
source("code/profile.R") # calculates counts per million (CPM) for each gene,
# and plots the distribution of CPM values for each sample
source("code/deg_list.R") # selects differential expressed genes (DEGs) based on input cut-offs,
# outputs an annotated tables with DEGs to a txt file
```

4.1 Introduction

The code chuck below describes processing of raw fastq files and mapping reads to the *Bifidobacterium longum* subsp. *infantis* ATCC 15697 transcriptome. The following software is required (can be installed to a conda environment):

1. FastQC (v0.11.9)
2. Cutadapt (v3.4)

3. Bowtie2 (v2.4.4)
4. Kallisto v0.46.2
5. MultiQC (v1.11)
6. Parallel (v20210222)

Note: due to size limitation, raw fastq files and indexed references could not be stored in the GitHub repo. Thus, you will need to download the fastq files from the Gene Expression Omnibus, under accession instert_accession. Downloaded fastq files should be in `data/fastq/`. The reference fasta files should be in `data/refs/`. Sample names should be in `data/runids.txt`.

Alternatively, you can run `code/qc_readmapping.sh` instead of the code chunk below.

4.2 Summary of the script

1. Quality control of raw reads was carried out using FastQC
2. Illumina sequencing adapters and short reads (< 20 bp) were removed using Cutadapt
3. Reads were mapped against rRNA and tRNA gene sequences extracted from the *Bifidobacterium longum* subsp. *infantis* ATCC 15697 genome (GenBank accession no. CP001095.1) using Bowtie2. Unmapped (filtered) reads were saved and used further
4. Filtered reads were mapped to the *Bifidobacterium longum* subsp. *infantis* ATCC 15697 (CP001095.1) using Kallisto
5. The quality of raw/filtered reads, as well as the results of Bowtie2/Kallisto mapping were summarized in `data/multiqc_report.html` generated via MultiQC

```
#!/bin/bash
echo $BASH_VERSION
set -ex

# required software: fastqc (v0.11.9), cutadapt (v3.4), bowtie2 (v2.4.4), kallisto (v0.46.2), multiqc (
# sample names should be in data/runids.txt

# activate conda enviroment with required software
eval "$(command conda 'shell.bash' 'hook' 2> /dev/null)" # initializes conda in sub-shell
conda activate transcriptomics
conda info|egrep "conda version|active environment"

# create directories
mkdir data/qc1 # qc results for raw reads
mkdir data/qc2 # qc results for filtered reads
mkdir data/fq_trim # trimmed reads
mkdir data/fq_filt # filtered reads
mkdir data/sam # sam files produced during bowtie2 alignment; will be deleted
mkdir data/kallisto # kallisto mapping results

# run fastqc on raw reads
cat data/runids.txt | parallel "fastqc data/fastq/{}.fastq.gz --outdir data/qc1"

# trim adapters using cutadapt
cat data/runids.txt | parallel "cutadapt -m 20 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTC \
-o data/fq_trim/{}.fastq.gz data/fastq/{}.fastq.gz \
&> data/fq_trim/{}.fastq.qz.log"

# filter reads mapping to rRNA and tRNA genes
```

```
# build bowtie2 index
bowtie2-build data/refs/Binfantis_ATCC15697_rRNA_tRNA.fasta data/refs/Binfantis_ATCC15697_rRNA_tRNA
# align reads via bowtie2; save ones that did not align to a separate file
cat data/runids.txt | parallel "bowtie2 -x data/refs/Binfantis_ATCC15697_rRNA_tRNA \
-U data/fq_trim/{}.fastq.gz \
-S data/sam/{}.sam \
--un data/fq_filt/{}.fastq \
&> data/fq_filt/{}.log"
rm -rf data/sam
cat data/runids.txt | parallel "gzip data/fq_filt/{}.fastq"

# run fastqc on filtered reads
cat data/runids.txt | parallel "fastqc data/fq_filt/{}.fastq.gz --outdir data/qc2"

# pseudolalign reads to transcriptome
# build kallisto index
kallisto index -i data/refs/Binfantis_ATCC15697_transcriptome.index data/refs/Binfantis_ATCC15697_trans
# map reads to indexed reference via kallisto
cat data/runids.txt | parallel "kallisto quant -i data/refs/Binfantis_ATCC15697_transcriptome.index \
-o data/kallisto/{} \
--single \
-l 200 \
-s 20 \
data/fq_filt/{}.fastq.gz \
&> data/kallisto/{}_2.log"

# run multiqc
export LC_ALL=en_US.utf-8
export LANG=en_US.utf-8
multiqc -d . -o data
```

---

5 Using R/bioconductor to import and analyze RNA-seq data

5.1 Importing count data into R

After read mapping with Kallisto, TxImport was used to read Kallisto outputs into the R environment.

```
# read the study design file
targets <- read_tsv("data/studydesign.txt")
# set file paths to Kallisto output folders with quantification data
files <- file.path("data/kallisto", targets$file_name, "abundance.tsv")
# check that all output files are present
all(file.exists(files))
```

```
## [1] TRUE
```

```
# use 'tximport' to import Kallisto output into R
txi_kallisto <- tximport(files,
                         type = "kallisto",
```

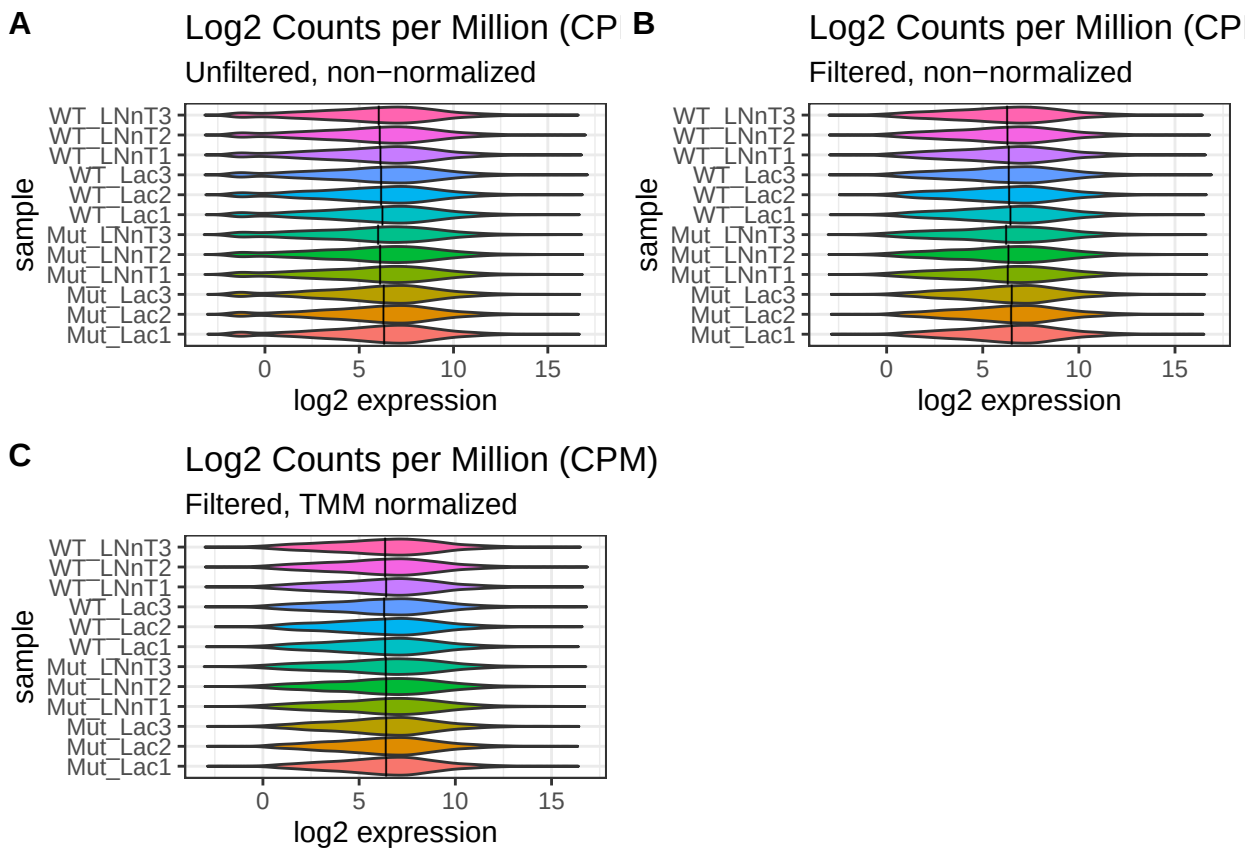
```
                          txOut = TRUE, # import at transcript level
                          countsFromAbundance = "lengthScaledTPM")

# capture variables of interest from the study design
condition <- as.factor(targets$condition)
condition <- factor(condition, levels = c("WT_Lac", "Mut_Lac", "WT_LNnT", "Mut_LNnT"))
batch <- as.factor(targets$batch)
strain <- as.factor(targets$strain)
carb <- as.factor(targets$carb)
# capture sample labels for later use
sampleLabels <- targets$sample

# saw a table with raw counts for GEO submission
raw_counts <- as.tibble(txi_kallisto$counts, rownames = "locus_tag")
colnames(raw_counts) <- c("geneID", sampleLabels)
write_tsv(raw_counts, "results/tables/GEO_kallisto_raw_counts.txt")

# use gt package to produce the study design table
gt(targets) %>%
  cols_align(
    align = "left",
    columns = TRUE
  )
```

sample	file_name	condition	batch	strain	carb
WT_Lac1	25-JCM-lac1_S53_L001_R1_001	WT_Lac	1	ATCC15697	Lac
Mut_Lac1	26-M3-lac1_S54_L001_R1_001	Mut_Lac	1	M3	Lac
WT_LNnT1	27-JCM-LNnT-1_S55_L001_R1_001	WT_LNnT	2	ATCC15697	LNnT
Mut_LNnT1	28-M3-LNnT-1_S56_L001_R1_001	Mut_LNnT	2	M3	LNnT
WT_Lac2	29-JCM-lac2_S57_L001_R1_001	WT_Lac	1	ATCC15697	Lac
Mut_Lac2	30-M3-lac2_S58_L001_R1_001	Mut_Lac	1	M3	Lac
WT_LNnT2	31-JCM-LNnT-2_S59_L001_R1_001	WT_LNnT	2	ATCC15697	LNnT
Mut_LNnT2	32-M3-LNnT-2_S60_L001_R1_001	Mut_LNnT	2	M3	LNnT
WT_Lac3	33-JCM-lac3_S61_L001_R1_001	WT_Lac	1	ATCC15697	Lac
Mut_Lac3	34-M3-lac3_S62_L001_R1_001	Mut_Lac	1	M3	Lac
WT_LNnT3	35-JCM-LNnT-3_S63_L001_R1_001	WT_LNnT	2	ATCC15697	LNnT
Mut_LNnT3	36-M3-LNnT-3_S64_L001_R1_001	Mut_LNnT	2	M3	LNnT

5.2 Filtering and normalization

```
myDGEList <- DGEList(txi_kallisto$counts)
# plot unfiltered, non-normalized CPM
p1 <- profile(myDGEList, sampleLabels, "Unfiltered, non-normalized")
# filter counts
cpm <- cpm(myDGEList)
keepers <- rowSums(cpm>1)>=3 # only keep genes that have cpm>1 (== not zeroes) in more than 2 samples (
myDGEList.filtered <- myDGEList[keepers,]
# plot filtered, non-normalized CPM
p2 <- profile(myDGEList.filtered, sampleLabels, "Filtered, non-normalized")
# normalize counts via the TMM method implemented in edgeR
```

```
myDGEList.filtered.norm <- calcNormFactors(myDGEList.filtered, method = "TMM")
# plot filtered, normalized CPM
p3 <- profile(myDGEList.filtered.norm, sampleLabels, "Filtered, TMM normalized")
# compare distributions of the CPM values
plot_grid(p1, p2, p3, labels = c('A', 'B', 'C'), label_size = 12)
```



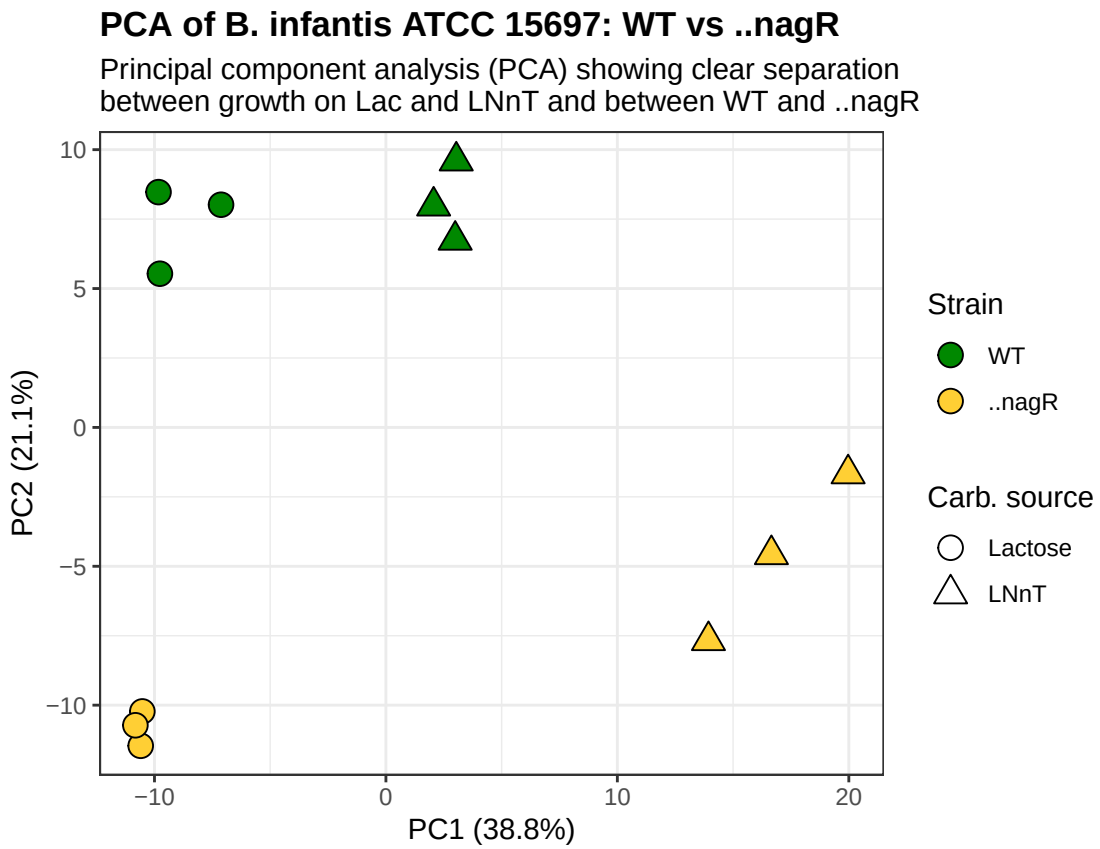
Filtering was carried out to remove lowly expressed genes. Genes with less than 1 count per million (CPM) in at least 3 or more samples filtered out. This procedure reduced the number of genes from **2508** to **2366**. In addition, the TMM method was used for between-sample normalization.

5.3 PCA plot

Principal Component Analysis (PCA) plots reduce complex datasets to a 2D representation where each axis represents a source of variance (known or unknown) in the dataset. As you can see from the plots below, Principal Component 1 (PC1; X-axis), which accounts for >38% of the variance in the data, is separating the samples based on carbon source. PC2 (Y-axis) accounts for a smaller source of variance (~21%) and can be attributed to variation between strains of *Bifidobacterium longum* subsp. *infantis* ATCC 15697: WT and *ΔnagR*.

```
# running PCA
log2.cpm.filtered.norm <- cpm(myDGEList.filtered.norm, log=TRUE)
pca.res <- prcomp(t(log2.cpm.filtered.norm), scale.=F, retx=T)
pc.var <- pca.res$sdev^2 # sdev^2 captures eigenvalues from the PCA result
pc.per <- round(pc.var/sum(pc.var)*100, 1) # calculate percentage of the total variation explained by e
# converting PCA result into a tibble for plotting
```

```
pca.res.df <- as_tibble(pca.res$x)

# plotting PCA
ggplot(pca.res.df) +
  aes(x=PC1, y=PC2, label=sampleLabels, shape = carb, fill = strain) +
  geom_point(size=4) +
  scale_shape_manual(name = "Carb. source",
                     breaks=c("Lac","LNnT"),
                     values=c(21, 24),
                     labels=c("Lactose", "LNnT")) +
  scale_fill_manual(name = "Strain",
                     breaks=c("ATCC15697","M3"),
                     values=c("#008800", "#ffcf34"),
                     labels=c("WT", "ΔnagR")) +
  guides(fill = guide_legend(override.aes=list(shape=21))) +
  xlab(paste0("PC1 (",pc.per[1],"%",")")) +
  ylab(paste0("PC2 (",pc.per[2],"%",")")) +
  labs(title= "PCA of B. infantis ATCC 15697: WT vs ΔnagR",
       subtitle = "Principal component analysis (PCA) showing clear separation \nbetween growth on Lac a
       color = "strain", shape="carb") +
  coord_fixed(ratio=1.2) +
  theme_bw() +
  theme(plot.title = element_text(face="bold"))
```

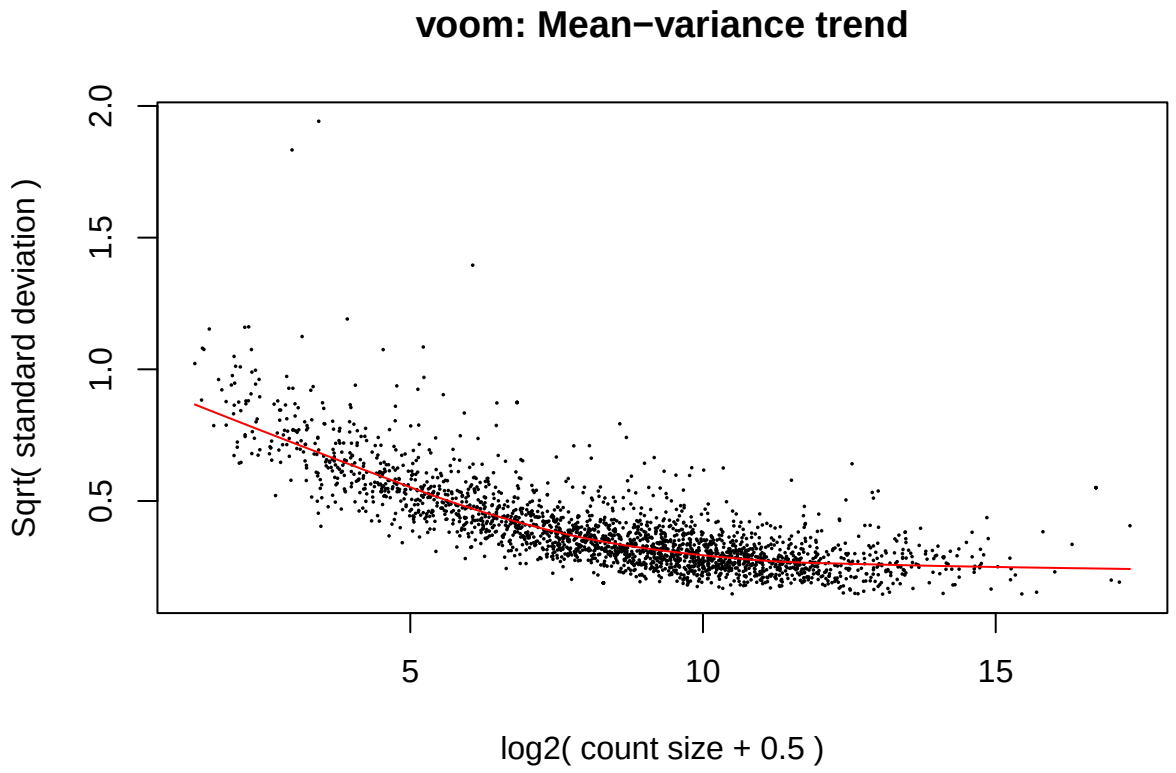


```
# save the figure as pdf
ggsave("results/figures/figure_2B.pdf", device = "pdf", width = 5, height = 5)
```

5.4 Differentially expressed genes

To identify differentially expressed genes (DEGs), precision weights were first applied to each gene based on its mean-variance relationship using VOOM. Linear modeling and bayesian stats were employed using Limma to find genes that were up- or down-regulated by 2-fold or more at false-discovery rate (FDR) of 0.01.

```
# setting up model matrix without intercept
design <- model.matrix(~0 + condition)
colnames(design) <- levels(condition)
# using VOOM function from Limma package to apply precision weights to each gene
v.DEGList.filtered.norm <- voom(myDGEList.filtered.norm, design, plot = TRUE)
```




```
fit <- lmFit(v.DEGList.filtered.norm, design)
# setting up contrast matrix for pairwise comparisons of interest
contrast.matrix <- makeContrasts(Mut_WT_Lac = Mut_Lac - WT_Lac,
                                 Mut_WT_LNnT = Mut_LNnT - WT_LNnT,
                                 LNnT_WT = WT_LNnT - WT_Lac,
                                 LNnT_Mut = Mut_LNnT - Mut_Lac,
                                 levels=design)
fits <- contrasts.fit(fit, contrast.matrix)
```

```
# extracting stats
ebFit <- eBayes(fits)
```

DEGs were annotated based on a RAST-annotated version of the *Bifidobacterium longum* subsp. *infantis* ATCC 15697 genome, which was additionally subjected to extensive manual curation performed in the web-based mcSEED environment, a private clone of the publicly available SEED platform. The manual curation focused on annotating genes encoding functional roles (transporters, glycoside hydrolases, downstream catabolic enzymes, transcriptional regulators) for bifidobacterial carbohydrate metabolism.

```
# create a master annotation table
seed.ann <- read_tsv('data/annotation/SEED_annotations.tsv')
corr <- read_tsv('data/annotation/Binfantis_ATCC15697_GenBank_vs_mcSEED.txt')
final.ann <- right_join(seed.ann, corr, by = c('seed_id' = 'seed_id')) %>%
  select(locus_tag, annotation)

# annotate DEGs
# Mut_Lac vs WT_Lac
myTopHits.Mut <- topTable(ebFit, adjust ="BH", coef=1, number=2600, sort.by="logFC")
deg_list(myTopHits.Mut, -1, 1, 0.01, "results/tables/DEG_Mut_Lac_vs_WT_Lac.txt")
```

locus_tag	annotation
Blon_0879	Predicted N-acetyl-glucosamine kinase 2, ROK family (EC 2.7.1.59)
Blon_0881	Glucosamine-6-phosphate deaminase (EC 3.5.99.6)
Blon_0882	N-acetylglucosamine-6-phosphate deacetylase (EC 3.5.1.25)
Blon_2347	Type II HMOs transporter (Blon_2347) I, substrate-binding protein
Blon_2341	hypothetical protein
Blon_2344	Type II HMOs transporter (Blon_2344) II, substrate-binding protein
Blon_2346	Type II HMOs transporter, permease protein 1
Blon_2343	Type II HMOs transporter, permease protein 1
Blon_2345	Type II HMOs transporter, permease protein 2
Blon_2342	Type II HMOs transporter, permease protein 2
Blon_1132	hypothetical protein
Blon_2352	Predicted HMO transporter Blon_2352, substrate-binding protein
Blon_2183	PTS system, glucose-specific IIABC (EC 2.7.1.69) @ PTS system, fructose-specific IIABC (EC 2.7.1.202)
Blon_1498	hypothetical protein
Blon_2349	N-acetylneuraminate lyase (EC 4.1.3.3)
Blon_2177	Lacto-N-biose and Galacto-N-biose ABC transporter 1, periplasmic substrate-binding protein @ Type I HMO
Blon_2351	Predicted HMO transporter Blon_2351, substrate-binding protein
Blon_2444	Maltose/maltodextrin ABC transporter, substrate binding periplasmic protein MalE
Blon_1192	hypothetical protein
Blon_0883	Lacto-N-biose and Galacto-N-biose ABC transporter 2, periplasmic substrate-binding protein
Blon_1198	NA
Blon_1480	ABC transporter, substrate-binding protein
Blon_0139	4-alpha-glucanotransferase (amylomaltase) (EC 2.4.1.25)
Blon_2176	Lacto-N-biose and Galacto-N-biose ABC transporter 1, permease component 1 @ Type I HMOs transporter,
Blon_1200	N-formylglutamate deformylase (EC 3.5.1.68)
Blon_1251	hypothetical protein
Blon_2442	Maltose/maltodextrin ABC transporter, permease protein MalF
Blon_2350	Predicted HMO transporter Blon_2350, substrate-binding protein
Blon_1831	putative lysin
Blon_2359	ABC transporter, permease component 2
Blon_1246	hypothetical protein

Blon_2441	Maltose/maltodextrin ABC transporter, permease protein MalG
Blon_2361	ABC transporter, ATP-binding protein
Blon_0884	Lacto-N-biose and Galacto-N-biose ABC transporter 2, permease component 1
Blon_2332	Lactose and galactose permease, GPH translocator family
Blon_1199	hypothetical protein
Blon_1244	hypothetical protein
Blon_0789	Sucrose specific transcriptional regulator CscR, LacI family
Blon_0786	ABC-type nitrate/sulfonate/bicarbonate transport system, permease component
Blon_0788	Sucrose permease, major facilitator superfamily
Blon_0787	Exo-beta-(2-1/2-6)-fructofuranosidase 2, GH32

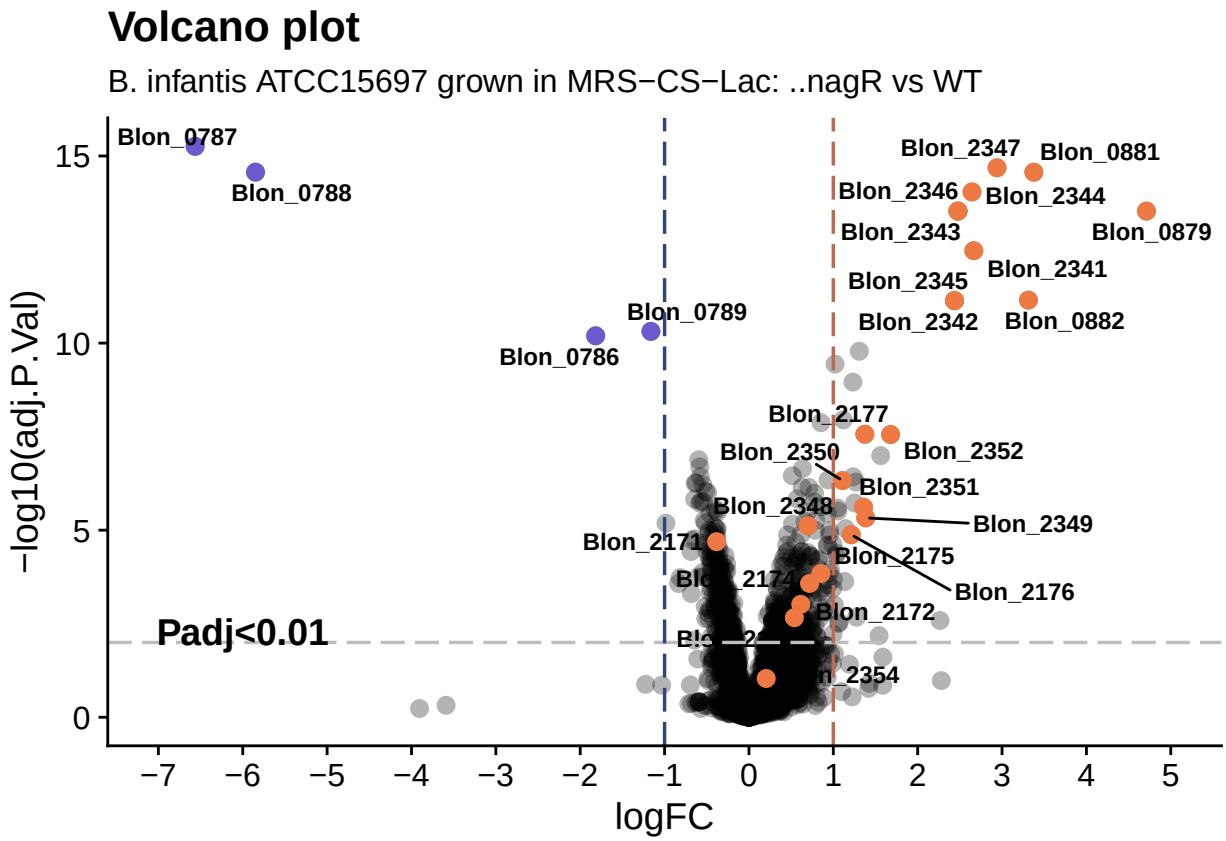
```
# annotate DEGs
# WT_LNnT vs WT_Lac
myTopHits.WT <- topTable(ebFit, adjust ="BH", coef=3, number=2600, sort.by="logFC")
deg_list(myTopHits.WT, -1, 1, 0.01, "results/tables/DEG_WT_LNnT_vs_WT_Lac.txt")
```

locus_tag	annotation
Blon_0879	Predicted N-acetyl-glucosamine kinase 2, ROK family (EC 2.7.1.59)
Blon_0881	Glucosamine-6-phosphate deaminase (EC 3.5.99.6)
Blon_0882	N-acetylglucosamine-6-phosphate deacetylase (EC 3.5.1.25)
Blon_2347	Type II HMOs transporter (Blon_2347) I, substrate-binding protein
Blon_2344	Type II HMOs transporter (Blon_2344) II, substrate-binding protein
Blon_2346	Type II HMOs transporter, permease protein 1
Blon_2343	Type II HMOs transporter, permease protein 1
Blon_2341	hypothetical protein
Blon_2349	N-acetylneuraminate lyase (EC 4.1.3.3)
Blon_2351	Predicted HMO transporter Blon_2351, substrate-binding protein
Blon_2345	Type II HMOs transporter, permease protein 2
Blon_2342	Type II HMOs transporter, permease protein 2
Blon_2352	Predicted HMO transporter Blon_2352, substrate-binding protein
Blon_2350	Predicted HMO transporter Blon_2350, substrate-binding protein
Blon_2177	Lacto-N-biose and Galacto-N-biose ABC transporter 1, periplasmic substrate-binding protein @ Type I HM
Blon_2475	Maltose/maltodextrin transport ATP-binding protein MalK (EC 3.6.3.19)
Blon_2348	HMO cluster exo-alpha-(2-3/2-6)-sialidase, GH33
Blon_0883	Lacto-N-biose and Galacto-N-biose ABC transporter 2, periplasmic substrate-binding protein
Blon_2176	Lacto-N-biose and Galacto-N-biose ABC transporter 1, permease component 1 @ Type I HMOs transporter,
Blon_2175	Lacto-N-biose and Galacto-N-biose ABC transporter 1, permease component 2 @ Type I HMOs transporter,
Blon_0884	Lacto-N-biose and Galacto-N-biose ABC transporter 2, permease component 1
Blon_2174	1,3-beta-galactosyl-N-acetylhexosamine phosphorylase (EC 2.4.1.211)
Blon_2173	N-acetylhexosamine 1-kinase (EC 2.7.1.162)
Blon_2172	UTP-hexose-1-phosphate uridylyltransferase involved in lacto-N-biose utilization, predicted
Blon_2064	Transcriptional regulator of galactose metabolism, DeoR family
Blon_1549	hypothetical protein
Blon_0787	Exo-beta-(2-1/2-6)-fructofuranosidase 2, GH32
Blon_0788	Sucrose permease, major facilitator superfamily
Blon_2332	Lactose and galactose permease, GPH translocator family
Blon_0885	Lacto-N-biose and Galacto-N-biose ABC transporter 2, permease component 2
Blon_0419	ABC-type anion transport system, duplicated permease component
Blon_0387	Riboflavin synthase eubacterial/eukaryotic (EC 2.5.1.9)
Blon_0389	6,7-dimethyl-8-ribityllumazine synthase (EC 2.5.1.78)
Blon_0388	GTP cyclohydrolase II (EC 3.5.4.25) / 3,4-dihydroxy-2-butanone 4-phosphate synthase (EC 4.1.99.12)
Blon_0153	NA

5.5 Volcano plot: *Bifidobacterium longum* subsp. *infantis* ATCC 15697 grown in MRS-CS-Lac: *ΔnagR* vs WT

Volcano plots are convenient ways to represent gene expression data because they combine magnitude of change (X-axis) with significance (Y-axis). Since the Y-axis is the inverse log10 of the adjusted Pvalue, higher points are more significant. In the case of this particular plot, there are many genes in the upper right of the plot, which represent genes that are significantly **upregulated** in the *ΔnagR* mutant grown in MRS-CS-Lac, compared to WT grown in MRS-CS-Lac.

```
# list stats for all genes in the dataset to be used for making volcano plot
myTopHits <- topTable(ebFit, adjust ="BH", coef=1, number=2600, sort.by="logFC")
myTopHits.df <- myTopHits %>%
  as_tibble(rownames = "geneID")
# select only genes with significant logFC and adj.P.Val
myTopHits.df.de <- subset(myTopHits.df, (logFC > 1 | logFC < -1) & adj.P.Val < 0.01)
# create a vector containing locus_tags of genes predicted to be in the NagR regulon
targets.nagR <- c("Blon_0879", "Blon_0881", "Blon_0882", "Blon_2171", "Blon_2172","Blon_2173", "Blon_21
# create a vector containing locus_tags of genes predicted to be in the CscR regulon
targets.cscR <- c("Blon_0789", "Blon_0788", "Blon_0787", "Blon_0786")
# subset volcano plot data based targets.nagR and targets.cscR
myTopHits.nagR <- subset(myTopHits.df, geneID %in% targets.nagR)
myTopHits.cscR <- subset(myTopHits.df, geneID %in% targets.cscR)
# subset data labels(NagR regulon) for volcano plot
myTopHits.df$nagR <- myTopHits.df$geneID
myTopHits.nagR_selected <- myTopHits.df$nagR %in% myTopHits.nagR$geneID
myTopHits.df$nagR[!myTopHits.nagR_selected] <- NA
# subset data labels(CscR regulon) for volcano plot
myTopHits.df$cscR <- myTopHits.df$geneID
myTopHits.cscR_selected <- myTopHits.df$cscR %in% myTopHits.cscR$geneID
myTopHits.df$cscR[!myTopHits.cscR_selected] <- NA

# create the volcano plot
ggplot(myTopHits.df) +
  aes(y=-log10(adj.P.Val), x=logFC, text = paste("Symbol:", geneID)) +
  geom_point(size=3, shape = 16, color="black", alpha=.3) +
  geom_point(mapping=NULL, myTopHits.nagR, size = 3, shape = 16, color= "sienna2", inherit.aes = TRUE) +
  geom_point(mapping=NULL, myTopHits.cscR , size = 3, shape = 16, color= "slateblue", inherit.aes = TRU
  geom_text_repel(aes(label = nagR), size = 3, fontface=2, color="black") +
  geom_text_repel(aes(label = cscR), size = 3, fontface=2, color="black") +
  geom_hline(yintercept = -log10(0.01), linetype="longdash", colour="grey", size=0.6) +
  geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=0.6) +
  geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=0.6) +
  annotate("text", x=-6, y=-log10(0.01)+0.3,
           label=paste("Padj<0.01"), size=5, fontface="bold") +
  scale_x_continuous(limits=c(-7,5), breaks = -7:5) +
  labs(title="Volcano plot",
       subtitle = "B. infantis ATCC15697 grown in MRS-CS-Lac: ΔnagR vs WT") +
  theme(plot.title = element_text(face="bold")) +
  theme_cowplot()
```

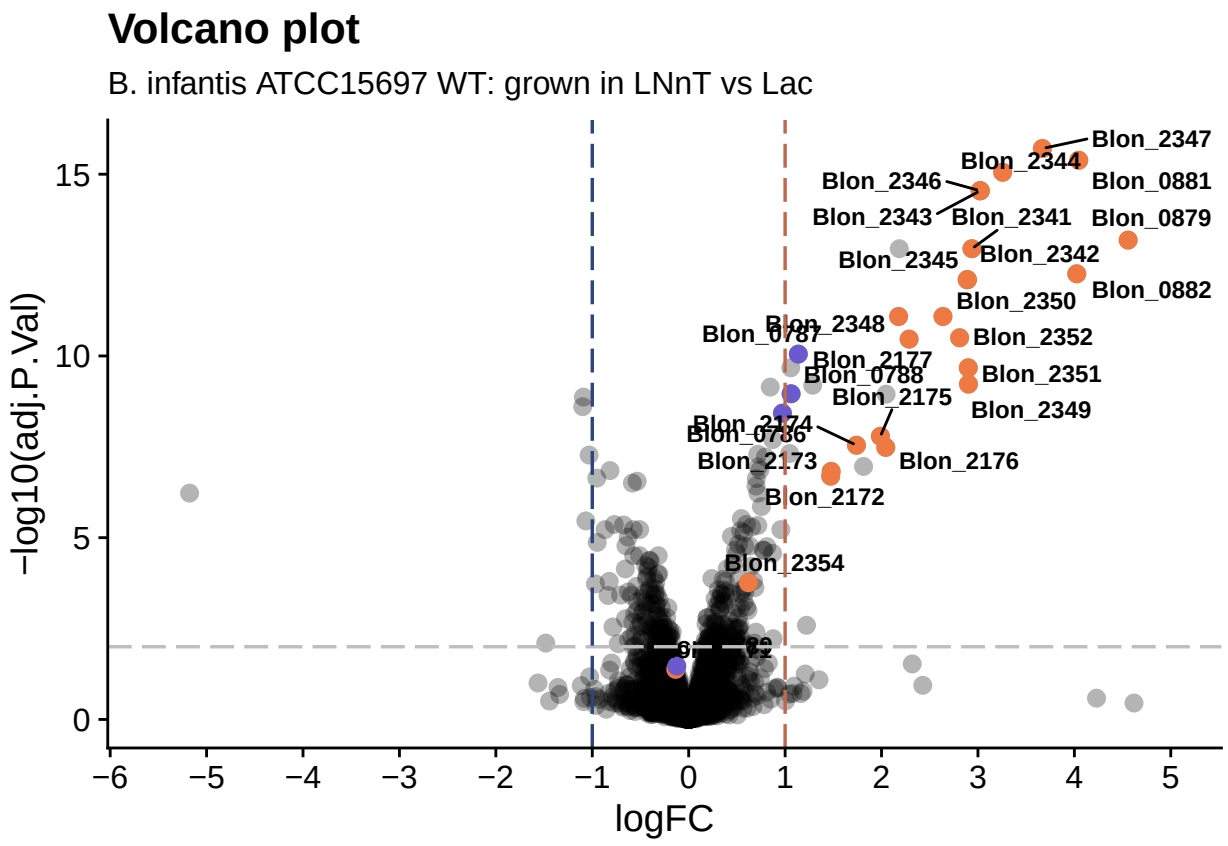



```
# save the figure as pdf
ggsave("results/figures/figure_2C.pdf", device = "pdf", width = 8, height = 5)
```

5.6 Volcano plot: *Bifidobacterium longum* subsp. *infantis* ATCC 15697 WT: grown in MRS-CS-LNnT vs grown in MRS-CS-Lac

```
# listing stats for all genes in the dataset to be used for making volcano plot
myTopHits3 <- topTable(ebFit, adjust ="BH", coef=3, number=2600, sort.by="logFC")
myTopHits.df3 <- myTopHits3 %>%
  as_tibble(rownames = "geneID")
# select only genes with significant logFC and adj.P.Val
myTopHits.df3.de <- subset(myTopHits.df3, (logFC > 1 | logFC < -1) & adj.P.Val < 0.01)
# subset volcano plot data based targets.nagR and targets.cscR
myTopHits.nagR3 <- subset(myTopHits.df3, geneID %in% targets.nagR)
myTopHits.cscR3 <- subset(myTopHits.df3, geneID %in% targets.cscR)
# subset volcano plot data labels (NagR-controlled genes)
myTopHits.df3$nagR <- myTopHits.df3$geneID
myTopHits.nagR_selected3 <- myTopHits.df3$nagR %in% myTopHits.nagR3$geneID
myTopHits.df3$nagR[!myTopHits.nagR_selected3] <- NA
# subset volcano plot data labels (CscR-controlled genes)
myTopHits.df3$cscR <- myTopHits.df3$geneID
myTopHits.cscR_selected3 <- myTopHits.df3$cscR %in% myTopHits.cscR3$geneID
myTopHits.df3$cscR[!myTopHits.cscR_selected3] <- NA
```

```
# create a volcano plot
ggplot(myTopHits.df3) +
  aes(y=-log10(adj.P.Val), x=logFC, text = paste("Symbol:", geneID)) +
  geom_point(size=3, shape = 16, color = "black", alpha =.3) +
  geom_point(mapping=NULL, myTopHits.nagR3, size = 3, shape = 16, color = "sienna2", inherit.aes = TRUE
  geom_point(mapping=NULL, myTopHits.cscR3 , size = 3, shape = 16, color = "slateblue", inherit.aes = TR
  geom_text_repel(aes(label = nagR), size = 3, fontface=2, color="black", max.overlaps = 100) +
  geom_text_repel(aes(label = cscR), size = 3, fontface=2, color="black") +
  geom_hline(yintercept = -log10(0.01), linetype="longdash", colour="grey", size=0.6) +
  geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=0.6) +
  geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=0.6) +
  annotate("text", x=-6, y=-log10(0.01)+0.3,
           label=paste("Padj<0.01"), size=5, fontface="bold") +
  scale_x_continuous(limits=c(-5.5,5), breaks = -6:5) +
  labs(title="Volcano plot",
       subtitle = "B. infantis ATCC15697 WT: grown in LNnT vs Lac") +
  theme(plot.title = element_text(face="bold")) +
  theme_cowplot()
```




```
# save the figure as pdf
ggsave("results/figures/figure_2D.pdf", device = "pdf", width = 8, height = 5)
```

5.7 Heatmap

Heatmap was plotted using the pheatmap package. Rows are clustered by hierarchical clustering. Data are scaled by Z-score for each row.

```
colnames(v.DEGList.filtered.norm$E) <- sampleLabels
diffGenes <- v.DEGList.filtered.norm$E
diffGenes.df <- as_tibble(diffGenes, rownames = "geneID")
diffGenes.df.two_conditions <- subset(diffGenes.df, geneID %in% myTopHits.df.de$geneID | geneID %in% myT
    relocate(geneID, WT_Lac1, WT_Lac2, WT_Lac3, Mut_Lac1, Mut_Lac2, Mut_Lac3, WT_LNnT1, WT_LNnT2, WT_LNn
diffGenes.two_conditions <- as.matrix(diffGenes.df.two_conditions[,-1])
rownames(diffGenes.two_conditions) <- diffGenes.df.two_conditions$geneID
# plot the heatmap
pheatmap(diffGenes.two_conditions,
         scale = "row",
         cluster_rows = F,
         cluster_cols = F,
         angle_col = 45,
         gaps_col = c(3, 6, 9),
         cellwidth = 10,
         cellheight = 10,
         filename = "results/figures/figure_S3.pdf")
```

5.8 Session info

The output from running 'sessionInfo' is shown below and details all packages and version necessary to reproduce the results in this report.

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] pheatmap_1.0.12    ggrepel_0.9.1      cowplot_1.1.1      matrixStats_0.61.0
##  [5] edgeR_3.36.0       limma_3.50.0       gt_0.3.1           tximport_1.22.0
##  [9] forcats_0.5.1      stringr_1.4.0      dplyr_1.0.7        purrr_0.3.4
## [13] readr_2.1.1        tidyr_1.1.4        tibble_3.1.6       ggplot2_3.3.5
## [17] tidyverse_1.3.1    knitr_1.37         tinytex_0.36       rmarkdown_2.11
##
## loaded via a namespace (and not attached):
```

```
##  [1] httr_1.4.2         bit64_4.0.5        vroom_1.5.7        jsonlite_1.7.2
##  [5] modelr_0.1.8       assertthat_0.2.1   highr_0.9          cellranger_1.1.0
##  [9] yaml_2.2.1         pillar_1.6.4       backports_1.4.1    lattice_0.20-45
## [13] glue_1.6.0         digest_0.6.29      RColorBrewer_1.1-2 checkmate_2.0.0
## [17] rvest_1.0.2        colorspace_2.0-2   htmltools_0.5.2    pkgconfig_2.0.3
## [21] broom_0.7.11       haven_2.4.3        scales_1.1.1       tzdb_0.2.0
## [25] farver_2.1.0       generics_0.1.1     ellipsis_0.3.2     withr_2.4.3
## [29] cli_3.1.0          magrittr_2.0.1     crayon_1.4.2       readxl_1.3.1
## [33] evaluate_0.14      fs_1.5.2           fansi_0.5.0        xml2_1.3.3
## [37] tools_4.1.2        hms_1.1.1          lifecycle_1.0.1    Rhdf5lib_1.16.0
## [41] munsell_0.5.0      reprex_2.0.1       locfit_1.5-9.4     compiler_4.1.2
## [45] rlang_0.4.12       rhdf5_2.38.0       grid_4.1.2         rhdf5filters_1.6.0
## [49] rstudioapi_0.13    labeling_0.4.2     gtable_0.3.0       DBI_1.1.2
## [53] R6_2.5.1           lubridate_1.8.0    fastmap_1.1.0      bit_4.0.4
## [57] utf8_1.2.2         stringi_1.7.6      parallel_4.1.2     Rcpp_1.0.7
## [61] vctrs_0.3.8        dbplyr_2.1.1       tidyselect_1.1.1   xfun_0.29
```